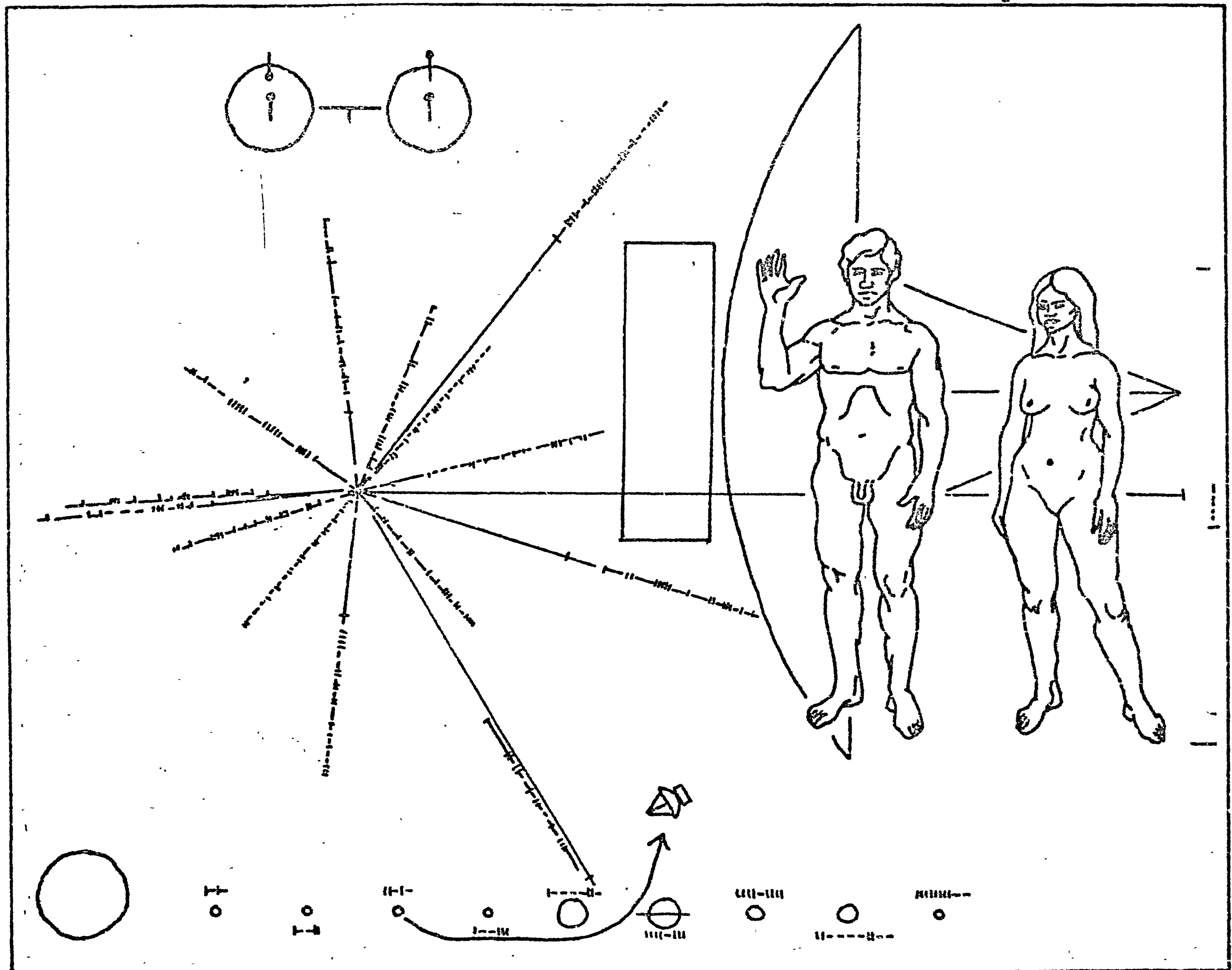
UNIVERSITE PARIS 8

U.E.R. INFORMATIQUE 808-96-70 POSTE 297 ROUTE DE LA TOURELLE 75571 PARIS CEDEX 12



LISP T 1600

Prof. Dr. H. Stoyan
Universität Erlangen-Nürnberg
Institut für Mathematische Maschinen
und Datenverarbeitung (Informatik VIII)
Am Weichselgarten 9
91058 Erlangen

Manuel de Référence Provisoire

INSTITUT DE L'ENVIRONNEMENT
14-20, rue Erasme
75005 – PARIS
Téléphone : 325-42-61

Patrick GREUSSAY

Février 1975

TABLE DES MATIERES

.../...

TABLE DES MATIERES (fin)

INTRODUCTION.

Le système LISP T 1600 est conçu pour la recherche et l'enseignement en Informatique musicale et en Intelligence Artificielle. Sa facilité d'emploi et sa clarté en font un excellent outil d'enseignement. L'incorporation de fonctions puissantes, et sa possibilité d'emploi en mode conversationnel en font, malgré les limitations ou taille mémoire, l'outil le plus utilisé pour la recherche au sein du Groupe d'Intelligence Artificielle de Vincennes.

Ce manuel n'est pas réellement destiné aux débutants, mais aux utilisateurs ayant déjà une certaine expérience de LISP. Il décrit en principe tous les aspects courants du système. Ce manuel n'engage l'auteur que sur les principes de base, et sera sujet à des remaniements, au fur et à mesure de l'évolution normale du système et de ses utilisateurs.

Si, au cours d'un travail, vous détectez une erreur manifeste du système, ou une situation très anomalique, laissez tout dans l'état, et venez me chercher, çà permettra de faire sauter les erreurs qui subsistent (inévitablement) dans le système.

Je crois à propos de remercier J. CHAILLOUX, Giuseppe ENGLERT, Harald WERTZ, J.C. HALGAND, P.L. NEUMAN, J. POINDRON, J.E. SCHOETTL, J.P. BOUDIER, S. CHARALAMBIDES, D. GOOSSENS, qui ont lu la version préparatoire du manuel et dont les excellentes suggestions ont et auront influencé la conception du système dont ce manuel est le manuel.

Jean ZEITOUN et Alain BUIS ont permis et encouragé la construction du LISP T 1600 à l'Institut de l'Environnement, en y créant une atmosphère dans laquelle la conception et la mise au point de logiciels avancés est un véritable plaisir.

.../...

GENERALITES.

L'interprete tourne en mode EVAL : il lit un atome ou un appel de fonction, l'évalue, imprime le résultat de l'évaluation, et recommence. Quelque chose comme çà :

```
(WHILE T
       (PRINT(EVAL(READ))))
```

On peut entrer des S-expressions sur cartes ou au clavier de la machine à écrire (MAE). Sur cartes, on peut écrire en format libre sur toute la carte. Au clavier, la MAE indique qu'elle attend un ligne en tapant "?", on tape alors à la suite.

FORME DES ATOMES :

. çà peut être un nombre, i.e. une suite de chiffres, éventuellement précédée de "-". Une suite de chiffres précédée de "+" n'est pas considérée comme un nombre.

. çà peut être un atome non-numérique, toutes les combinaisons de caractères sont bonnes, sauf les séparateurs.

SEPARATEURS :

"espace", "." , "(" , ")" , ";". On peut constituer toutefois des atomes comprenant des séparateurs grâce à la fonction READCH. Un macro-caractère constitue un nouveau séparateur.

COMMENTAIRES :

Tout se qui ce trouve entre deux ";" sera ignoré, quoique imprimé si c'est sur cartes.

NOMBRES :

LISP T 1600 n'utilise que les nombres entiers.

.../...

Ils devront être inclus dans l'intervalle [-32768 , 32767].
Les atomes numériques ne comportent ni C, ni P-valeurs ; ils sont stockés directement dans les listes ou dans les variables.
Des nombres hexadécimaux peuvent être utilisés directement, ils ont la représentation : CCCC HH , les "C" étant des chiffres hexadécimaux.

A l'impression, sauf modification (voir MOT DE CONTROLE) tous les objets édités (par PRIN1 ou PRINT) seront précédés automatiquement d'un espace.

Sauf modification, le nombre d'atomes définis par l'utilisateur ne pourra dépasser 200 (nombres non-inclus).

Sauf modification, l'utilisateur dispose de 2.600 doublets libres. L'utilisateur peut savoir à tout instant combien il reste de doublets libres en évaluant

(CDR 'REM)

FORME INTERNE DES ATOMES :

Un atome non-numérique est (approximativement) représenté en mémoire ainsi :

C-valeur	P-valeur
PNAME	

La partie PNAME contient la sortie de caractères qui représente son nom externe (imprimable).

ATTENTION :

A la lecture, seuls les 8 premiers caractères d'un atome seront pris en compte, le reste sera ignoré.

.../...

La partie C-valeur contiendra, à tout instant, la valeur de l'atome.
La partie C-valeur sera considérée comme le CAR de l'atome.
Les C-valeurs des atomes définis par l'utilisateur seront initialisées à la valeur "indéfini", ce qui provoquera à l'évaluation l'erreur A8.
La C-valeur des SUBRS contient l'adresse du code-machine correspondant.
Certains atomes du système contiennent leur propre adresse en C-valeur, c'est le cas de : NIL,QUOTE,LAMBDA,EXPR,FEXPR,T.
Nul besoin, donc de les quoter.

exemple : (LIST NIL QUOTE LAMBDA EXPR FEXPR T)
donnera : (NIL QUOTE LAMBDA EXPR FEXPR T)

La partie P-valeur contiendra la P-liste de l'atome. Elle sera considérée comme le CDR de l'atome. Les fonctions DE et DF ne touchent pas aux C-valeurs des atomes. On peut donc redéfinir momentanément une fonction standard comme une EXPR, une NEXPR, ou une FEXPR (voir TYPES DE FONCTIONS).

exemple : rédéfinition de SETQ pour qu'elle imprime à l'appel :
"nom de la variable = sa valeur"

```
(DF SETQ (1L)
    (SET (PRIN1 (CAR 1L)) (PROGN
         (PRIN1 =) (PRINT (EVAL (CADR 1L))) )))
```

l'appel : (SETQ X 5) provoquera alors l'impression
"X = 5"

Je puis annuler cette redéfinition en évaluant (RRLACD 'SETQ NIL)

<u>MACRO-CARACTERES</u> :

Il y a la possibilité <u>à la lecture</u> d'appeler une fonction LISP à la seule occurence d'un caractère. Le résultat de l'appel de cette fonction sera substitué à la place du caractère lu.

.../...

On procède ainsi :

(MCHAR caractère fonction) : ceci dit que désormais, la fonction sera appelée avec NIL comme argument, si le caractère est lu. Le caractère "'" est une macro standard. Si elle ne l'était pas, on aurait défini :

```
(MCHAR '(LAMBDA () (LIST QUOTE (READ)))   .
```

Et, à la présentation de 'ob, ob étant un atome ou une liste, sera substitué : (QUOTE ob).
Le macro-caractère ainsi défini est alors considéré comme un nouveau séparateur.

ATTENTION :

Un macro-caractère particulier ne peut être défini qu'une seule fois.

S-EXPRESSIONS POINTEES :

Elles sont acceptées par le système, sous la forme la plus générale

(expression.expression)

exemple : ((A.(B.C)).(D E . (F G H . I)))

EVALUATION DES VARIABLES :

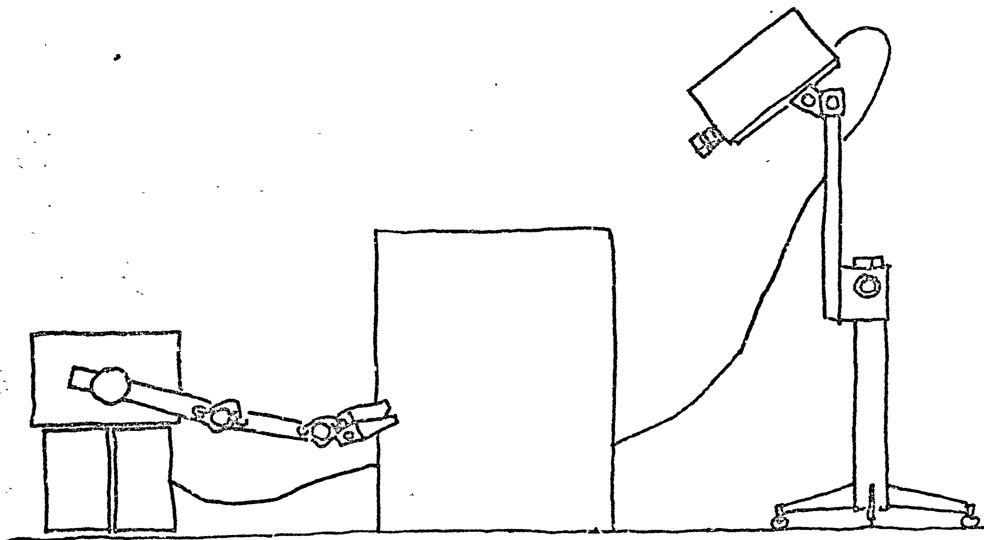
Il n'y a pas de A-liste. A tout moment, la valeur des variables est accessible dans la C-valeur. A l'entrée de PROGs ou à l'appel de fonctions dont la variable est argument formel, la valeur d'une variable est empilée dans la pile de travail et restituée à la sortie (voir LISTING DE L'INTERPRETE) au moyen des fonctions BIND et UNBIND.
Ce mode d'accès est très efficient, mais ne permet plus l'usage des FUNARG commeen LISP 1.5. Reste que nous n'avons pratiquement jamais eu d'occasion sérieuse de les utiliser.

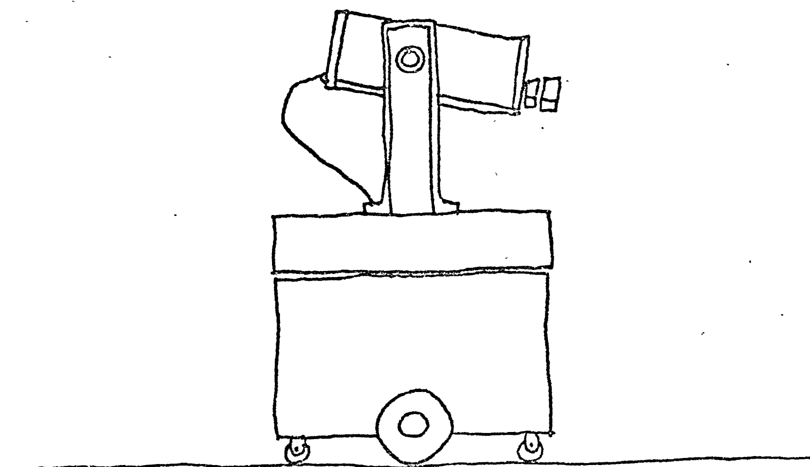
.../...

Un bon ensemble de traces, d'éditeurs, de pretty-print, et d'indexs-catalogues est disponible, écrit directement en LISP.

Le système effectue pour son compte le strict minimum de CONS. Les garbage collections dus au système seront donc relativement peu fréquents.

Le système est défini indépendemment de la T 1600, et peut donc être transféré rapidement sur un autre ordinateur.





TYPES DES FONCTIONS.

Le système comporte six types de fonctions :

SUBR,NSUBR,FSUBR,EXPR,NEXPR,FEXPR.

Les SUBR sont des fonctions standard en langage-machine.
Les EXPR sont des fonctions définies en LISP par l'utilisateur.

SUBR, EXPR : les arguments, en nombre fixé sont évalués, puis sont passés à la fonction.

NSUBR, NEXPR : les arguments, en nombre quelconque, sont évalués. Puis ces valeurs sont rassemblées en une liste qui est passée à la fonction.

FSUBR, FEXPR : les arguments, en nombre quelconque, ne sont pas évalués, mais sont rassemblés en une liste qui est passée à la fonction.

Si une fonction (standard ou utilisateur) n'a pas assez d'arguments fournis à l'appel, les arguments manquants sont supposés être NIL. La fonction ne peut donc pas distinguer entre NIL donné comme argument, et pas d'argument du tout. C'est fort commode.

exemple : (FOO) et (FOO NIL) reviennent au même.
(SETQ X NIL) a le même effet que (SETQ X).

Si l'appel de la fonction comporte trop d'arguments (pour les EXPR et SUBR), ils sont évalués mais ignorés par la fonction. Donc, si on a une fonction f à m arguments, et qu'on évalue (f x1 x2 ... x_n), avec n>m, les arguments x_{n+1} , x_{n+2} , ... , x_m seront ignorés.

.../...

Les arguments formels en trop jouent donc le même rôle que les variables locales des PROG, et sont initialisés à NIL.

On dispose d'une fonction standard PROGN, à nombre quelconque d'arguments. Celle-ci évalue ses arguments de gauche à droite et retourne la valeur du dernier.
Les LAMBDA-expressions ont un PROGN implicite, i.e.

(LAMBDA (A1 A2) e_1 e_2 ... e_n)

sera intérprêté comme :

(LAMBDA (A1 A2) (PROGN e_1 e_2 ... e_n)

et la valeur ramenée sera celle de e_n.

Les NSUBR sont les fonctions standard

LIST , PLUS et TIMES.

Les NEXPR sont des fonctions dont la LAMBDA-expression a un CADR atomique.

exemple : dans

((LAMBDA X e_1 e_2 ... e_n) a_1 a_2 ... a_m)

X sera lié à la liste des valeurs de a_1 a_2 ... a_m.

exemple : si LIST n'était pas une fonction standard, elle serait définie ainsi :

(DE LIST L L)

En général, on définira une NEXPR ainsi :

(DE nom atome e_1 e_2 ... e_n)
la
fonction

Le COND est généralisé de telle sorte que chaque clause peut comprendre n⩾1 expressions.

.../...

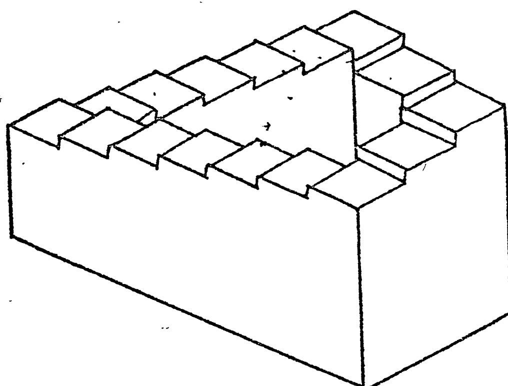
exemple :

```
(COND
    (P1     E11     E12     E13)
    (P2     E21     E22)
    (P3)
    (P4     E41))
```

sera considéré comme équivalent à :

```
(COND
    (P1     (PROGN E11     E12     E13))
    (P2     (PROGN E21     E22))
    ((SETQ x P3) x)
    (P4     E41)
    (T      NIL))
```

En lISP T 1600, VRAI est équivalent à non-NIL.
Ceci permet de considérer une fonction comme ramenant une valeur utile, et pour tester une certaine condition. Par exemple (MEMQ x y) ramène, ou bien NIL, ou bien le CDR de y qui commence par x. De même la valeur de OR est la valeur de son premier argument vrai, i.e. non-NIL, et la valeur de AND est, ou bien NIL, ou bien la valeur de son dernier argument.



MOT DE CONTROLE.

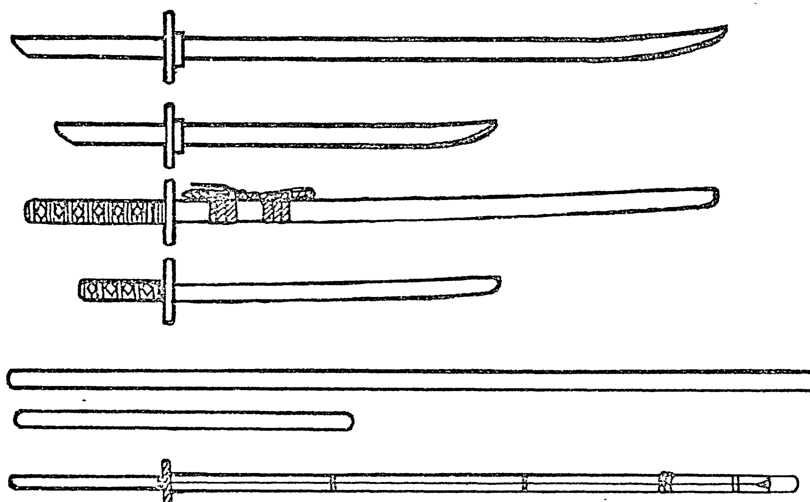
Il existe un mot de 16 bits qui contrôle certains états du système. Ce mot est accessible au moyen des fonctions SETBIT et CLRBIT.

BITS	FONCTION DU POSITIONNEMENT
15	= 0 : un espace est placé AVANT tout objet, atome ou liste, imprimé dans une ligne.
	= 1 : suppression de cet espace
14	= 0 : sortie télétype
	= 1 : sortie imprimante
13	= 0 : les macros-caractères sont pris en compte
	= 1 : il n'en est rien
12	= 0 : sortie de nombres ou décimal
	= 1 : sortie en héxadécimal
11	= 0 : les ";" sont pris en compte
	= 1 : non
10	= 0 : les "/" sont pris en compte en entrée
	= 1 : non
9	= 0 : ne restitue pas les "/" en sortie

.../...

BITS	FONCTION DU POSITIONNEMENT
8	= 0 : impression en TOP-LEVEL = 1 : non-impression TOP-LEVEL
7	= 0 : listing carte lue sur LP = 1 : listing carte lue sur TTY
6	= 0 : rien = 1 : lecture sur disque
5	= 0 : rien = 1 : écriture sur disque
0	= 0 : lecture sur télétype = 1 lecture sur lecteur de cartes

Au chargement de LISP tous ces bits sont à zéro.
Toute erreur (voir ERREURS) remet le mot de contrôle à zéro.



ERREURS.

S'il y a une erreur detectée, LISP tape "*ER type" et repasse la main au clavier et à la boucle de lecture de l'interprête. Aucune définition de fonction, ou attribution de valeur à une variable n'est modifiée, et une garbage-collection est effectuée dans la foulée.
Voici les types des erreurs :

LC : erreur de lecture. Cà peut être une liste commençant par ")" ou bien "." non suivi de liste atome ")". En général, çà détecte trop de ")".

RT : l'utilisateur a tenté de faire un RETURN sans être dans un PROG.

FS : la pile de travail est débordée. Erreur en général due à une récursion infinie.

FM : il n'y a plus de doublets disponibles en zone liste.

AT : il y a trop d'atomes définis par l'utilisateur.

A2 : fonction non-définie, détectée dans APPLY. Le nom de cette fonction inconnue est IMPRIME avant le diagnostic.

A6 : GO ou GOTO à une étiquette absente du PROG courant. Le nom de l'étiquette est imprimé.

A8 : tentative d'évaluation d'une variable non-initialisée. Le nom de la variable est imprimé. En général, la variable a été simplement oubliée.

A9 : fonction non-définie, détectée dans EVAL. Le nom de la fonction inconnue est imprimé.

.../...

FONCTIONS DE CONTROLE.

(EVAL ob) → évalue ob et ramène sa valeur.
SUBR

(APPLY f l) → applique la fonction f à la liste l d'arguments évalués. Ramène la valeur résultante de cette application. La fonction f ne doit pas être une FSUBR ni une FEXPR.
SUBR

(PRØGN e_1 e_2...e_n) → évalue en séquence les expressions e_1,...e_n et ramène en valeur, la valeur de e_n.
FSUBR

(EPRØGN l) → évalue en séquence les éléments de la liste l, et ramène en valeur celle du dernier élément de l.
SUBR

(EVLIS l) → l étant une liste (e_1 e_2...e_n), EVLIS ramène en valeur
(val[e_1] val[e_2] ... val[e_n])
SUBR

(AND e_1 e_2...e_n) → les expressions e_1 e_2...e_n sont évaluées de gauche à droite jusqu'à ce que la valeur d'un des e_i soit NIL : dans ce cas, la valeur du AND est NIL.
Si aucun des e_i ne donne NIL, la valeur du AND est celle de e_n.
FSUBR

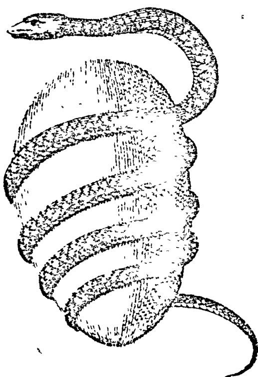
(ØR e_1 e_2...e_n) → les expressions e_1 e_2...e_n sont évaluées de gauche à droite. La valeur du OR est celle du premier e_i dont la valeur n'est pas NIL. Si tous les e_i sont NIL, la valeur du OR est NIL.
FSUBR

.../...

(WHILE e_1 e_2 ... e_n) → les expressions e_1 e_2 ... e_n sont évaluées de gauche à droite tant que la valeur de e_1 n'est pas NIL. WHILE boucle ainsi jusqu'à ce que la valeur de e_1 soit NIL. WHILE ramène alors la valeur NIL.
FSUBR

(CØND c_1 c_2 ... c_K) → CØND comporte un nombre quelconques d'arguments c_1 c_2 ... c_K appelés <u>clauses</u>.
FSUBR

Chaque clause c_i est une liste :
(l_1i l_2i ... l_ni) de $n \geqslant 1$ expressions.
Les clauses de CØND sont traitées en séquence comme suit :
La première expression l_1i est évaluée, ou bien comme <u>fausse</u> si = NIL, ou bien comme <u>vraie</u> si ≠ NIL.
Si la valeur de e_i1 est vraie, les expressions l_2i ... l_ni qui suivent dans la clause c_i sont évaluées en séquence, et la valeur de CØND est la valeur de l_ni, la dernière expression de la clause.
En particulier, si n = 1 i.e. si la clause c_i ne comporte qu'<u>une</u> expressions, la valeur du CØND est alors celle de l_1i.
Si e_1i, est fausse, alors le reste de la clause c_i est ignoré, et la clause suivante $c_i + 1$ est traitée.
Si aucun e_1i n'est vrai dans aucune clause, la valeur du CØND est NIL.



J'ai emprunté l'explication qui prècède au manuel du LISP BBN (Bolt, Beranek, Newman), en raison de sa remarquable clarté.

(PRØG l e_1 e_2 ... e_n) → l : liste de variables locales, ou NIL si y en a pas. e_1 ... e_n : suite d'expres-
FSUBR

.../...

sions dont chacune peut être précédée d'un atome qui servira d'étiquette aux GØ et GØTØ.

A l'entrée du PRØG, les variables locales sont réservées et initialisées à NIL. Puis les e_1 ... e_n sont évaluées en séquence, les étiquettes étant sautées. S'il n'y a pas de RETURN évalués, la valeur du PRØG est celle de e_n.

Les étiquettes consécutives sont autorisées, les étiquettes numériques sont interdites.

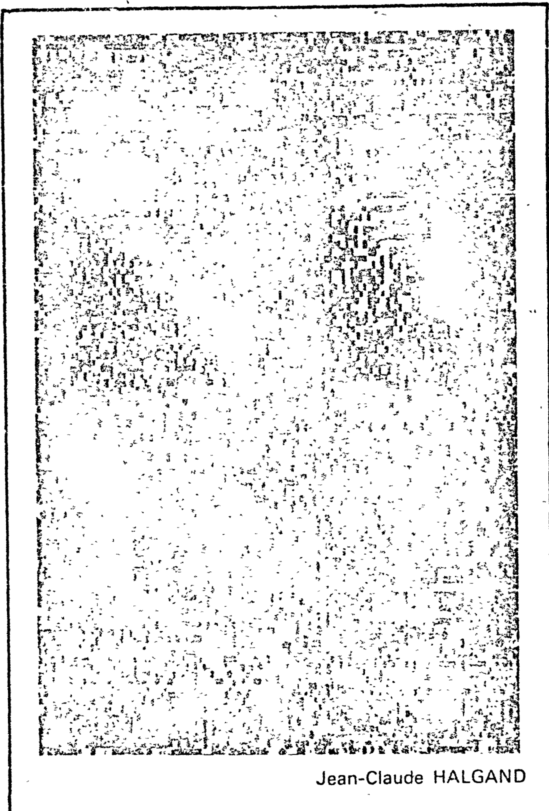
(RETURN ob) SUBR	→	sort du PRØG courant en ramenant la valeur de ob en valeur.
(GØ e) FSUBR	→	e : atome qui doit être une étiquette du PRØG courant. GØ fait reprendre l'évaluation du PRØG à partir de l'expression qui suit son étiquette.
(GØTØ e) SUBR	→	id. à GØ, mais l'expression e est évaluée, et doit donc ramener en valeur un atome qui est une étiquette du PRØG courant.
ATTENTIØN	:	GØ, GØTØ et RETURN peuvent tout à fait être évalués dans des fonctions appelées par un PRØG. Toutefois, sauf à utiliser un ESCAPE, on ne peut sortir d'un PRØG que par un RETURN, ou par la fin du PRØG, ou encore par une erreur (!).
(QUØTE ob) FSUBR	→	ramène ob sans l'évaluer. En LISP T 1600, on écrit plutôt 'ob.

.../...

(ESCAPE f e_1 ... e_n) → f : un atome non-numérique
e_1 ... e_n : une suite d'expressions à évaluer.
ESCAPE évalue les e_1 ... e_n en séquence. Si dans la portée du ESCAPE se produit l'évaluation de (f ob_1 ... ob_n), on évalue les ob_1 ... ob_n en séquence, et on sort du ESCAPE en ramenant en valeur celle de ob_n. Si celà ne se produit pas, la valeur du ESCAPE est celle de e_n.

NØTER QUE : ESCAPE est la fonction de saut la plus puissante compatible avec la structure de contrôle de LISP. Elle n'existe avec cette généralité qu'en LISP T 1600. Un ESCAPE est une sorte d'étiquette, et sa sortie une sorte de branchement ramenant une valeur.
Dans(ESCAPE EXIT e_1 e_2 ... e_n), par exemple, EXIT est lié avec une fonction interne nommée ESC. EXIT devient donc, dans la portée du ESCAPE, une fonction de sortie. Si, au cours de l'évaluation des e_1 e_2 ... e_n, on est amené à évaluer (EXIT ob_1 ... ob_n), on évalue les ob_1 ... ob_n et on sort du ESCAPE avec la valeur de ob_n.

Les ESCAPE peuvent s'imbriquer, et sauter les uns par dessus les autres.


Jean-Claude HALGAND

.../...

FONCTIONS D'ENTREE-SORTIE.

(READ) SUBR	→	Sa valeur est un objet lu, atome ou liste, sur carte ou au clavier de la MAE (voir MØT DE CØNTRØLE).
(PRINT ob) SUBR	→	imprime ob et ramène ob en valeur. L'impression est suivie d'un retour à la ligne, et précédée (sauf modification par (C (CLRBIT 15)) d'un espace.
(PRIN1 ob) SUBR	→	id. à PRINT, mais ob est seulement EDITE dans la ligne. Il n'y a pas d'impression de la ligne, elle sera après coup imprimée par l'appel d'un PRINT ou d'un TERPRI.
(TERPRI) SUBR	→	provoque un passage à la ligne (ce qui peut provoquer l'impression d'une ligne éditée), et ramène NIL en valeur.
(TTAB n) SUBR	→	le prochain objet édité (par PRINT ou PRIN1) sera placé à la position n dans la ligne TTAB ramène n en valeur.
(SPACES n) SUBR	→	laisse dans une ligne en cours d'édition, n espaces entre le dernier objet édité et le prochain objet édité dans la ligne. SPACES ramène n en valeur.
ATTENTIØNS	:	. si la somme des longueurs des objets édités dans une ligne par des PRIN1 ou des PRINT excède la longueur de ligne standard, un retour à la ligne automatique est provoqué par le système, et l'impression se poursuit à la ligne suivante.

.../...

. le contenu des cartes lues est systématiquement imprimé sur la MAE lors de la lecture. Pour inhiber provisoirement l'impression, positionner la clé 0 du pupitre. Enlever la clé 0 pour rétablir l'impression.

. si au clavier de la MAE, on demande l'évaluation d'un atome, taper après l'atome un espace avant d'aller à la ligne.

. avec (SETBIT 0) tout est lu sur cartes.
avec (CLRBIT 0) tout est lu au clavier de la MAE.

(READCH) SUBR → lit comme un atome le premier carctère immédiatement disponible dans le buffer de lecture et le ramène en valeur.
Si on arrive en fin de buffer, un nouvel enregistrement est automatiquement lu.

NOTER QUE : tout caractère, séparateur inclus, peut être "quoté" en le faisant précéder du caractère "/".

<u>exemple</u> : "AN/(NIE/)"

sera lu comme l'atome de 7 caractères

"AN(NIE)"

bien entendu, le "/" peut se quoter lui-même.

(PAGE) SUBR → provoque un saut de page sur l'imprimante.

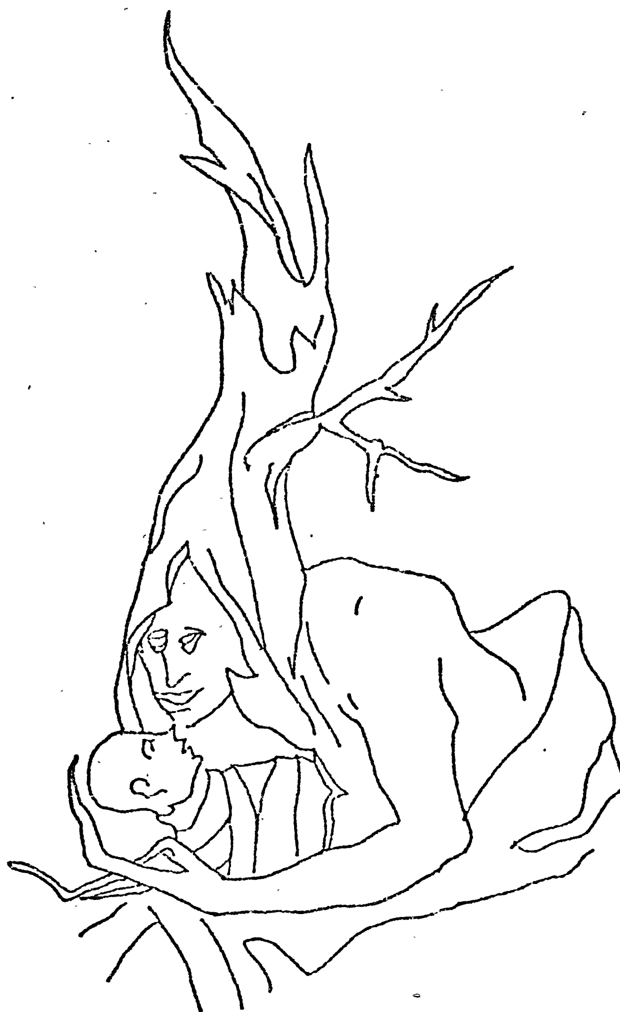
.../...

FONCTIONS DE DONNEES.

(ATØM x) SUBR	→	{ T si x est un atome { NIL sinon NIL est, pour ATØM, considéré comme un atome donc : (ATØM NIL) → T
(CAR x) SUBR	→	{ si x est un atome : sa C-valeur { si x est une liste : son 1er élément
(CDR x) SUBR	→	{ si x est un atome : sa P-liste { si x est une liste : x SANS son 1er élément
(CAAR x) (CADR x) (CDAR x) (CDDR x) (CADDDR x) (CAADR x) (CADDDR x) (CADADR x) (CAADDR x) (CADADDR x) SUBR		sont disponibles
(EQ x y) SUBR	→	si x et y sont des atomes { T si x = y { NIL sinon si x et y sont des listes { T si x et y sont PHYSIQUEMENT la même liste { NIL sinon

.../...

(NEQ x y) SUBR	→	(NØT(EQ x y))
(NULL x) SUBR	→	T si x = NIL NIL sinon
(NØT x) SUBR	→	identique à NULL
(MAPC l f) SUBR	→	applique la fonction f aux CARs successifs de la liste l et ramène NIL en valeur.



.../...

(CØNS x y)
SUBR

→ . si y est une liste, ramène une liste y avec x placé DEVANT l'ancien 1er élément de y

. si y est un atome, ramène la paire pointée (x . y)

(LIST e_1 ... e_n)
NSUBR

→ ramène la liste des valeurs des e_1 ... e_n

(LENGTH l)
SUBR

→ ramène le nombre d'éléments de la liste l

(NTH n l)
SUBR

→ n : un nombre, et l : une liste.
NTH ramène la partie de l qui commence par le nième élément.

<u>par exemple</u> :

si n = 1, la valeur est l
si n = 2, la valeur est (cdr l)
si n = 3, la valeur est (CDDR l)
etc...

(MEMQ ob l)
SUBR

→ ob : un atome, et l : une liste.
MEMQ ramène

{ NIL si ob $\notin$ l
{ sinon la partie de l qui commence par ob.

<u>par exemple</u> :

(MEMQ x_3 '(x_1 x_2 x_3 x_4))→(x_3 x_4)

(SETQ ob e)
FSUBR

→ donne à l'atome ob la valeur de l'expression e, et ramène celle-ci en valeur.

.../...

(SET e_1 e_2) SUBR	→	id. à SETQ, mais e_1 est évalué. A la valeur ramenée près, l'effet est équivalent à celui de RPLACA.
(NCØNS l_1 l_2) SUBR	→	concatère physiquement l_1 et l_2 (qui doivent être des listes), et ramène l_1 en valeur.
(NCØNC1 l_1 e) SUBR	→	(NCONC l_1 (LIST e)) Fort utile pour placer un atome à la fin d'une liste.
(EQUAL e_1 e_2) SUBR	→	{ T si e_1 et e_2 sont identiques NIL sinon surtout utilisé pour tester l'égalité de deux listes.
(APPEND l_1 l_2) SUBR	→	ramène la concaténation d'une CØPIE de l_1 à l_2.
(LISTP e) SUBR	→	{ e si e est une liste NIL sinon
(ASSQ at l) SUBR	→	l doit être une liste de listes, et at doit être un atome. ASSQ ramène la sous-liste de l dont at est le 1er élément, sinon NIL.
(CASSQ at l) SUBR	→	ramène le CDR de la sous-liste de l dont at est le 1er élément.

exemple :

```
(ASSQ  D  ((B.C)(D.E)(A.C)))
ramène :   (D.E)
(CASSQ  D  ((B.C)(D.E)(A.C)))
ramène :    E
```

.../...

(REVERSE l) SUBR	→	ramène une copie inversée de la liste l.
(GENSYM) SUBR	→	ramène à chaque appel un <u>nouvel</u> atome du type : G000n <u>Exemple</u> : (GENSYM) → G0001 puis (GENSYM) → G0002 etc...
(SUBST e at ob) SUBR	→	e : expression quelconque at : atome ob : liste ou atome ramène une copie de ob telle que toutes les occurences de at sont remplacées par e.
NOTER QUE	:	(APPEND L) : donne une copie du top level de L (SUBST NIL NIL L) : donne une copie <u>complète</u> de la liste L.
(MAP l f) SUBR	→	applique la fonction f à la liste l et à ses CDR successifs, et ramène NIL en valeur.
(RPLACA ob_1 ob_2)	→	. si ob_1 est une liste RPLACA élimine son 1er élément et place ob_2 à la place. . si ob_1 est un atome, RPLACA donne à cet atome la valeur ob_2.

.../...

dans les deux cas, la valeur ramenée par RPLACA est ob_1.
A la valeur ramenée près, l'effet est équivalent à celui de SET.

(RPLACD ob_1 ob_2) SUBR → . si ob_1 est une liste, RPLACD remplace son CDR par ob_2.

. si ob_1 est un atome, RPLACD remplace sa P-liste par ob_2.

Dans les deux cas, la valeur ramenée par RPLACD est ob_1.

(PUT ob v ind) SUBR → ob : atome ou liste
ind. : atome

. si ob est un atome, place sur sa P-liste la valeur v sous l'indicateur ind.

. si ob est une liste, elle est considérée comme une P-liste, et v est placé sous l'indicateur ind.

La valeur ramenée par PUT est ob.

(GET ob ind) SUBR → si ob est un atome, on considère alors sa P-liste, si ob est une liste, elle même. GET ramène la valeur trouvée sous l'indicateur ind dans la P-liste ; si ind n'existe pas, GET ramène NIL.

NØTER : En LISP T 1600 les P-listes ont la structure :
($indic_1$ val_1 $indic_2$ val_2 ... $indic_n$ val_n)

.../...

Donc GET et PUT, en testant les indicateurs vont de deux en deux éléments.

(NEXTL 1) → 1 DØIT être une variable qui a pour valeur une liste.
FSUBR

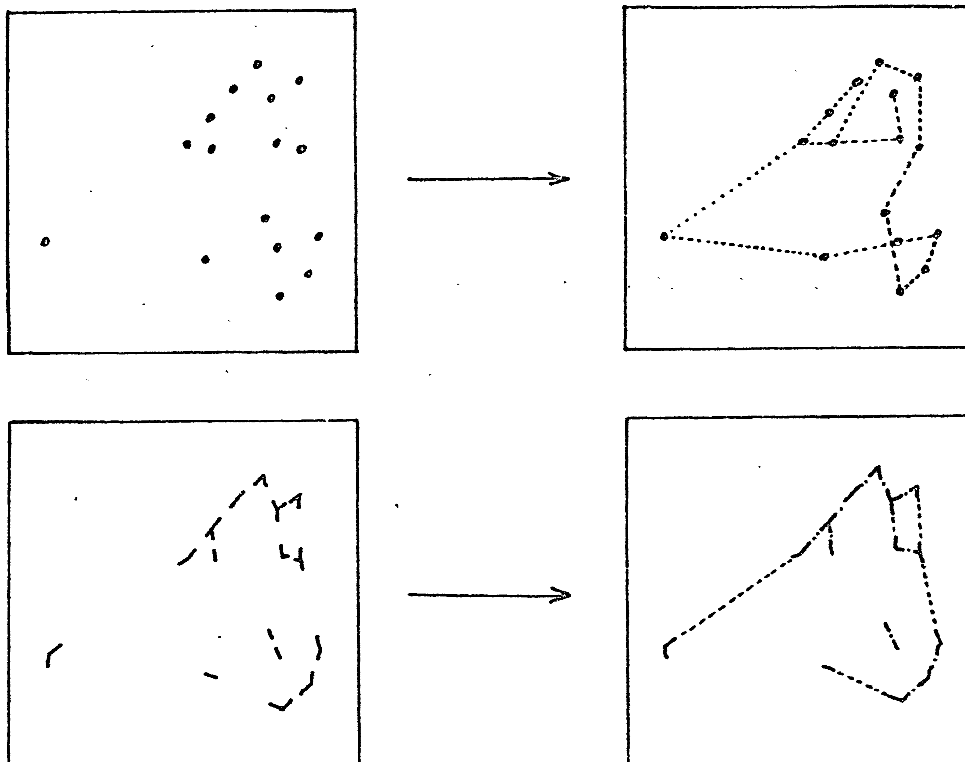
NEXTL ramène le CAR de la valeur de 1, et place dans 1 le CDR de sa valeur.

Exemple :

(SETQ X (NEXTL L))

est équivalent à :

(SETQ X (CAR L)) suivi de
(SETQ L (CDR L))



FONCTIONS D'ENREGISTREMENT.

(DE at 1 e_1 e_2 ... e_n) → permet de définir une fonction de nom at et de liste d'arguments 1. Cette fonction sera une EXPR. La valeur ramenée par DE est at.
FSUBR

(DF at 1 e_1 e_2 ... e_n) → id. à DE, mais la fonction définie sera FEXPR.
FSUBR

.DE place dans la P-liste de at, sous l'indicateur EXPR :
(LAMBDA 1 e_1 ... e_n).

.DF place dans la P-liste de at sous l'indicateur FEXPR :
(LAMBDA 1 e_1 ... e_n).

REMARQUE :
faire : (DE at 1 e_1 ... e_n) est équivalent
à faire : (PUT 'at '(LAMBDA 1 e_1 ... e_n)EXPR)
et faire : (DF at 1 c_1 ... e_n) est équivalent
à faire : (PUT 'at '(LAMBDA 1 e_1 ... e_n)FEXPR)

(MCHAR c f) → c : un caractère, et f : une fonction. Indique que désormais, si le caractère c est lu, la fonction f est appelée avec NIL comme argument. Le résultat de (f NIL) est alors placé dans la donnée lue à la place du carctère c. La fonction MCHAR permet donc de définir des fonctions qui seront appelées A LA LECTURE, par la seule présence du caractère c.
MCHAR ramène c en valeur.
FSUBR

.../...

FONCTIONS ARITHMETIQUES.

(ADD1 x) SUBR	→	x+1
(SUB1 x) SUBR	→	x-1
(DIFFER x y) SUBR	→	x-y
(QUØ x y) SUBR	→	x/y : division entière
(REM x y) SUBR	→	reste de x/y
(PLUS x_1 x_2 ... x_n) NSUBR	→	$x_1+x_2...+x_n$
(TIMES x_1 x_2 ... x_n) NSUBR	→	$x_1 * x_2 * ... * x_n$
(NUMBP x) SUBR	→	{ x si x est un nombre { NIL sinon
(ZERØP x) SUBR	→	{ 0 si x=0 { NIL sinon
(LT x y) SUBR	→	{ x si x<y { NIL sinon
(GT x y) SUBR	→	{ x si x>y { NIL sinon
(EQ x y) SUBR	→	{ T si x=y { NIL sinon

.../...

(NEQ x y) SUBR	→	T si x≠y NIL sinon
(GE x y) SUBR	→	x si x≥y NIL sinon
(LE x y) SUBR	→	x si x≤y NIL sinon
(GTZ x) SUBR	→	x si x>0 NIL sinon
(LØGAND x y) SUBR	→	ET binaire des nombres x et y
(LØGØR x y)	→	ØU binaire

exemples :

(LØGAND 0F00HH 35ABHH)	→	0500HH
(LØGØR 00FFHH FF00HH)	→	FFFFHH

(LØGSHIFT x n) → donne la valeur du nombre x décalé
- de n positions à droite si n>0
- de n positions à gauche si n<0

Exemple :

(LØGSHIFT 00FF0HH 4)	→	000FHH
(LØGSHIFT 00F0HH -4)	→	0F00HH

.../...

FONCTIONS DU SYSTEME.

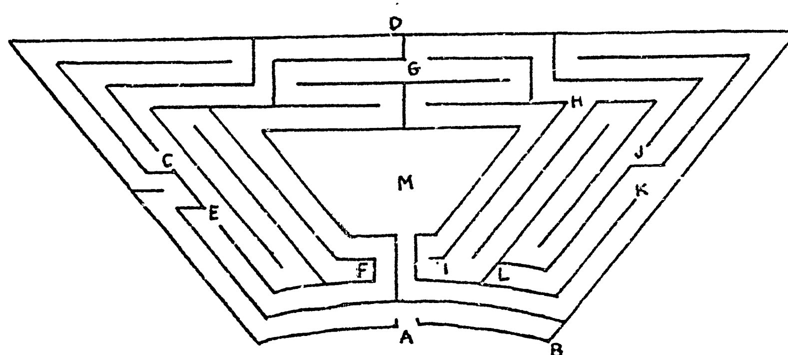
(RESET) SUBR → réinitialise le système LISP, et ramène un LISP frais. (RESET T) repasse la main à BØS/D.

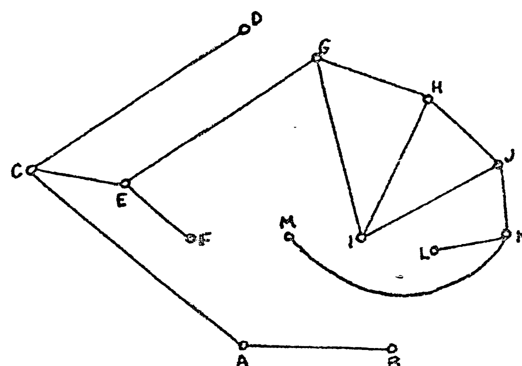
(SWITCH n) SUBR → n si la clé n ε [0,15] du pupitre est positionnée. NIL sinon

(ØBLIST) SUBR → ramène la liste de tous les atomes non-numériques introduits par l'utilisateur.

(SETBIT n) SUBR → place à 1 le but n ε [C,15] du mot de contrôle et ramène n en valeur.

(CLRBIT n) → place à 0 le bit n ε [0,15] du mot de contôle et ramène n en valeur.





FONCTIONS LØC et VAG.

NOTER :

La représentation des nombres en LISP T 1600 :
un nombre n est un doublet

ADNUMB	n

donc : soit l'adresse 1024 en mémoire :

1024	x

(CAR 1024)
NE DONNE PAS x , parce que 1024 c'est en fait :

ADNUMB	1024

Exemple : l'atome "FØØ" = supposons le précisément à l'adresse 1024 en mémoire.

(LØC 'FØØ) → 1024

donc (LØC objet) donne sa véritable adresse en mémoire sous la forme d'un nombre LISP

exemple : (ADD1(LØC'FØØ)) → 1025

INVERSEMENT :

Sachant que FØØ est en 1024
(VAG 1024) → FØØ

donc : (VAG adresse) donne la valeur LISP de l'objet stocké à cette adresse

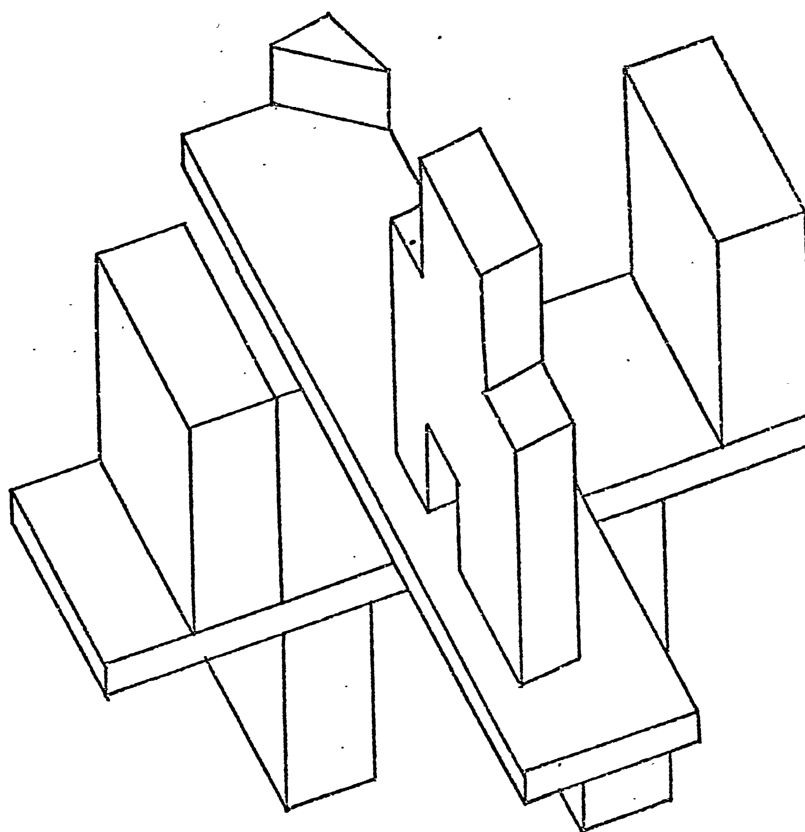
.../...

Ces fonctions sont très utiles pour modifier certains aspects du système, ex. : les limites de zones atome, mémoire libre, pile, etc.

exemple : pour obtenir 300 doublets supplémentaires en mémoire libre, faire :

```
(NULL(SET(CDR  'CAAR)(VAG(PLUS 600(LØC(CADR 'CAAR))))))
puis
(NULL(SET(CDR  'CADR)(VAG(PLUS 600(LØC(CADR 'CADR))))))
```

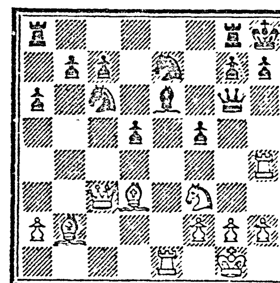
puis provoquer exprès une erreur lecture. Le système sera réinitialisé avec 300 doublets supplémentaires.



FONCTION GRAPHIQUE.

(CALCOMP x n) éxécute n fois sur la table traçante l'ordre x, traçant un segment de 1/10 mm.

x = 0 →
1 ↗
2 ↑
3 ↖
4 ←
5 ↙
6 ↓
7 ↘
8 lever la plume
9 baisser la plume


Les Blancs jouent et gagnent

NOTER :

Le système LØGØ-LISP permet de tracer en
- mode ordinaire incrémental
- mode x-y à coordonnées
- mode TØRTUE

ATTENTION :

Si vous voulez interrompre un traitement en cours parce que ça semble boucler, ou pour une autre raison, faites 4 appels console. BØS/D vous indique une erreur.
Tapez alors CLISP, et vous vous retrouvez dans la boucle de lecture LISP, avec vos programmes et vos données préservées.

.../...

NOTER :

un système d'entrées-sorties complet, ainsi que des primitives de gestion de fichier sont en cours de mise au point. Toutes les commandes FMS seront des fonctions standard du LISP T 1600.

Existent et sont disponibles :

- divers programmes de TRACE
- un sous-ensemble de l'éditeur INTERLISP
- des programmes de mise en page = PRETTY-PRINT
- un package d'indéxation de fonctions, écrit par Daniel G. BOBROW
- un LAP complet, donnant accès à toutes les instructions du langage machine de la T 1600
- un compilateur
- une version restreinte du langage CONNIVER
- une version du langage LØGØ : gestion de tortues.

Ces programmes seront documentés dans la prochaine version du manuel référence.

Toutefois voici les manipulations provisoirement nécessaires pour charger des fichiers LISP stockés sur disque :

sous BØS/D faire :

```
* CALL FUP3
* FDUPLI,LISP2,D4,LISP2,D2
*FDUPLI,nom du fichier,D4,nom du fichier,D2
```

Puis

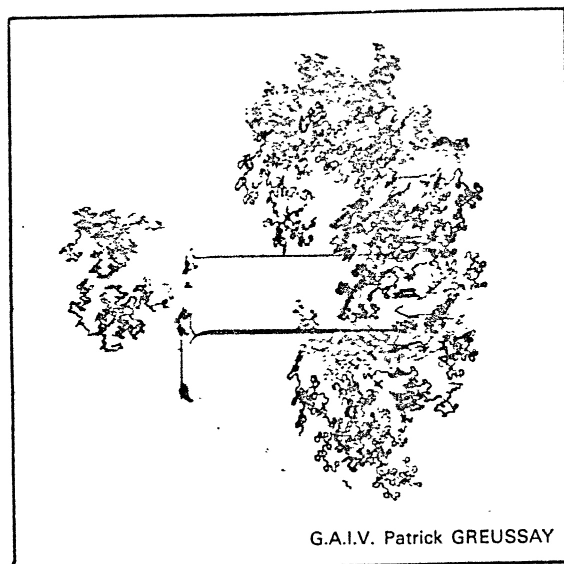
```
* JOB nom,,D2
* RUN LISP2
* LISP
```

.../...

LISP tape alors le "?"
Mettez la clé 0, puis tapez
(ØPENØLD nom du fichier)
et le fichier est alors chargé.

Les "noms de fichiers" peuvent être :

TRACEF:D
PRETTY:D
EDITØR:D
CØNNIV:D
LØGØTØ:D
LAPLAP:D
etc...


G.A.I.V. Patrick GREUSSAY

```
(SETBIT 14)
 14
'(TEST LISP T1600)
 (TEST LISP T1600)
(WHILE T (PRINT (EVAL (READ)))
         (MAPC '(________ ________ ________ ________ ________) 'PRIN1)
         (TERPRI))
NIL
 NIL
 ________ ________ ________ ________ ________
()
 NIL
 ________ ________ ________ ________ ________
(; COMMENTAIRE ;)
 NIL
 ________ ________ ________ ________ ________
(RPLACA 'SAVE (CADR 'ATOM)) ;ADRESSE DERNIER ATOME DEFINI DS (CADR 'ATOM) ;
 SAVE
 ________ ________ ________ ________ ________
1
 1
 ________ ________ ________ ________ ________
-1
 -1
 ________ ________ ________ ________ ________
FFFFHH
 -1
 ________ ________ ________ ________ ________
03FFHH
 1023
 ________ ________ ________ ________ ________
(SETBIT 11)
 11
 ________ ________ ________ ________ ________
';;;;
 ;;;;
 ________ ________ ________ ________ ________
(CLRBIT 11)     ; SI BIT 11 , INSENSIBLE AUX ;
 11
 ________ ________ ________ ________ ________
(SETBIT 12)     ; SI BIT 12 , SORTIE HEXA DES NOMBRES ;
 000CHH
 ________ ________ ________ ________ ________
-1
 FFFFHH
 ________ ________ ________ ________ ________
1023
 03FFHH
 ________ ________ ________ ________ ________
(CLRBIT 12)
 12
 ________ ________ ________ ________ ________
(SETQ X '/)/./(//)
 ).(/
 ________ ________ ________ ________ ________
X
 ).(/
 ________ ________ ________ ________ ________
'(SETBIT 9)     ; SI BIT 9 , SENSIBLE AUX '/' EN SORTIE ;
 9
 ________ ________ ________ ________ ________
X
```

```
/)/./(//
_________ _________ _________ _________ _________
(CLRBIT 9)
 9
_________ _________ _________ _________ _________
(SETBIT 13)     ; SI BIT 13 , INSENSIBLE AUX MACRO-CARACTERES ;
 13
_________ _________ _________ _________ _________
(SETQ X (QUOTE '))
 '
_________ _________ _________ _________ _________
X
 '
_________ _________ _________ _________ _________
(CLRBIT 13)
 13
_________ _________ _________ _________ _________
'/'                 ; INSENSIBLE AUX MACRO-CARACTERES SI QUOTES PAR / ;
 '
_________ _________ _________ _________ _________
(PROGN (PRIN1 'B) (PRIN1 'O) (PRINT 'N))
 B O N
 N
_________ _________ _________ _________ _________
(SETBIT 15)     ; SI BIT 15 PAS DE BLANCS ENTRE LES EDITIONS ;
15
__________________________________________________
(PROGN  (PRIN1 'B) (PRIN1 'O)  (PRINT 'N))
BON
N
__________________________________________________
(CLRBIT 15)
 15
_________ _________ _________ _________ _________
'A
 A
_________ _________ _________ _________ _________
''A
 (QUOTE A)
_________ _________ _________ _________ _________
'''A
 (QUOTE (QUOTE A))
_________ _________ _________ _________ _________
'(A)
 (A)
_________ _________ _________ _________ _________
(ADD1 1)
 2
_________ _________ _________ _________ _________
'(ADD1 1)
 (ADD1 1)
_________ _________ _________ _________ _________
'(A . (B . (C . D)))
 (A B C . D)
_________ _________ _________ _________ _________
'(((A . B) . C) . D)
 (((A . B) . C) . D)
_________ _________ _________ _________ _________
(COND (NIL) (T 1))
 1
_________ _________ _________ _________ _________
(COND (1))
```

```
1
_________ _________ _________ _________ _________
(COND (T 1 2 3))
3
_________ _________ _________ _________ _________
((LAMBDA L L) 'A 'B 'C)
(A B C)
_________ _________ _________ _________ _________
(APPLY 'CAR '((A B C)))
A
_________ _________ _________ _________ _________
(SETQ X 'CDR)
CDR
_________ _________ _________ _________ _________
(APPLY X '((A B C)))
(B C)
_________ _________ _________ _________ _________
(SETQ L '((ADD1 0) (ADD1 1) (SUB1 4)))
((ADD1 0) (ADD1 1) (SUB1 4))
_________ _________ _________ _________ _________
(EVLIS L)
(1 2 3)
_________ _________ _________ _________ _________
(EPROGN L)
3
_________ _________ _________ _________ _________
(ATOM)
T
_________ _________ _________ _________ _________
(ATOM 'A)
T
_________ _________ _________ _________ _________
(ATOM 1)
T
_________ _________ _________ _________ _________
(ATOM (OBLIST))
NIL
_________ _________ _________ _________ _________
(RPLACA 'X '(B O N))
X
_________ _________ _________ _________ _________
(RPLACD 'X '(N O N))
X
_________ _________ _________ _________ _________
(CAR 'X)
(B O N)
_________ _________ _________ _________ _________
(CDR 'X)
(N O N)
_________ _________ _________ _________ _________
(RPLACA '(A B) 'C)
(C B)
_________ _________ _________ _________ _________
(RPLACD '(A B) '(C))
(A C)
_________ _________ _________ _________ _________
(CONS 1 2)
(1 . 2)
_________ _________ _________ _________ _________
(CONS 1 '(2 3))
(1 2 3)
_________ _________ _________ _________ _________
```

```
(OBLIST)
 (L D C A N O B ).(/ X ;;;; SAVE ________ T1600 LISP TEST)
 _________ _________ _________ _________ _________
(NOT T)
 NIL
 _________ _________ _________ _________ _________
(NULL T)
 NIL
 _________ _________ _________ _________ _________
(NULL NIL)
 T
 _________ _________ _________ _________ _________
(EQ 1 2)
 NIL
 _________ _________ _________ _________ _________
(EQ 1 1)
 T
 _________ _________ _________ _________ _________
(EQ 'A 'B)
 NIL
 _________ _________ _________ _________ _________
(EQ 'A 'A)
 T
 _________ _________ _________ _________ _________
(EQ (CDR X) (CDR X))
 T
 _________ _________ _________ _________ _________
(EQ '(N O N) '(N O N))
 NIL
 _________ _________ _________ _________ _________
(CDR 'REM)
 2600
 _________ _________ _________ _________ _________
(NEQ 1 2)
 T
 _________ _________ _________ _________ _________
(NEQ 1 1)
 NIL
 _________ _________ _________ _________ _________
(NEQ 'A 'B)
 T
 _________ _________ _________ _________ _________
(NEQ 'A 'A)
 NIL
 _________ _________ _________ _________ _________
(NEQ (CDR X) (CDR X))
 NIL
 _________ _________ _________ _________ _________
(NEQ '(N O N) '(N O N))
 T
 _________ _________ _________ _________ _________
(GTZ 0)
 NIL
 _________ _________ _________ _________ _________
(GTZ -1)
 NIL
 _________ _________ _________ _________ _________
(GTZ 1)
 1
 _________ _________ _________ _________ _________
(GTZ 10000)
 10000
```

```
---------- ---------- ---------- ---------- ----------
(CDR 'HOP)
 NIL
---------- ---------- ---------- ---------- ----------
(PUT 'HOP 1 'UN)
 HOP
---------- ---------- ---------- ---------- ----------
(GET 'HOP 'UN)
 1
---------- ---------- ---------- ---------- ----------
(GET 'HOP 'DEUX)
 NIL
---------- ---------- ---------- ---------- ----------
(CDR 'HOP)
 (UN 1)
---------- ---------- ---------- ---------- ----------
(PUT 'HOP 'ONE 'UN)
 HOP
---------- ---------- ---------- ---------- ----------
(CDR 'HOP)
 (UN ONE)
---------- ---------- ---------- ---------- ----------
(PUT '(A 1 B 2) 3 'C)
 (A 1 B 2 C 3)
---------- ---------- ---------- ---------- ----------
(GET '(A 1 B 2) 'B)
 2
---------- ---------- ---------- ---------- ----------
(PUT '(1 A 2 B) 'C 2)
 (1 A 2 C)
---------- ---------- ---------- ---------- ----------
(GET '(1 A 2 B) 2)
 B
---------- ---------- ---------- ---------- ----------
(TIMES 0 1)
 0
---------- ---------- ---------- ---------- ----------
(TIMES -1 -1)
 1
---------- ---------- ---------- ---------- ----------
(TIMES 1 2 3 4 5)
 120
---------- ---------- ---------- ---------- ----------
(PLUS -1 -1)
 -2
---------- ---------- ---------- ---------- ----------
(PLUS 1 2 3 4 5)
 15
---------- ---------- ---------- ---------- ----------
(ZEROP 0)
 0
---------- ---------- ---------- ---------- ----------
(ZEROP 1)
 NIL
---------- ---------- ---------- ---------- ----------
(DIFFER 3 5)
 -2
---------- ---------- ---------- ---------- ----------
(DIFFER 5 5)
 0
---------- ---------- ---------- ---------- ----------
(DIFFER -5 5)
```

```
-10
-------- -------- -------- -------- --------
(QUO 21 3)
7
-------- -------- -------- -------- --------
(QUO 21 5)
4
-------- -------- -------- -------- --------
(QUO 5 21)
0
-------- -------- -------- -------- --------
(REM 10 2)
0
-------- -------- -------- -------- --------
(REM 9 2)
1
-------- -------- -------- -------- --------
(REM 2 9)
2
-------- -------- -------- -------- --------
(NUMBP 1)
1
-------- -------- -------- -------- --------
(NUMBP 'A)
NIL
-------- -------- -------- -------- --------
(NUMBP (OBLIST))
NIL
-------- -------- -------- -------- --------
(GT 2 1)
2
-------- -------- -------- -------- --------
(GT 1 1)
NIL
-------- -------- -------- -------- --------
(GT 1 2)
NIL
-------- -------- -------- -------- --------
(GE 2 1)
2
-------- -------- -------- -------- --------
(GE 1 1)
1
-------- -------- -------- -------- --------
(GE 1 2)
NIL
-------- -------- -------- -------- --------
(LT 2 1)
NIL
-------- -------- -------- -------- --------
(LT 1 1)
NIL
-------- -------- -------- -------- --------
(LT 1 2)
1
-------- -------- -------- -------- --------
(LE 2 1)
NIL
-------- -------- -------- -------- --------
(LE 1 1)
1
-------- -------- -------- -------- --------
```

```
(LE 1 2)
 1
 ---------- ---------- ---------- ---------- ----------
(ADD1 1)
 2
 ---------- ---------- ---------- ---------- ----------
(SUB1 1)
 0
 ---------- ---------- ---------- ---------- ----------
(ADD1 -1)
 0
 ---------- ---------- ---------- ---------- ----------
(SUB1 -1)
 -2
 ---------- ---------- ---------- ---------- ----------
(PROGN (SETQ X '(A B C))
       (WHILE X (PRIN1 (NEXTL X))))
 A B C NIL
 ---------- ---------- ---------- ---------- ----------
(LENGTH '(1 2 3))
 3
 ---------- ---------- ---------- ---------- ----------
(LENGTH NIL)
 0
 ---------- ---------- ---------- ---------- ----------
(NTH 1 '(A B C))
 (A B C)
 ---------- ---------- ---------- ---------- ----------
(NTH 2 '(A B C))
 (B C)
 ---------- ---------- ---------- ---------- ----------
(NTH 3 '(A B C))
 (C)
 ---------- ---------- ---------- ---------- ----------
(MAPC '(A B C) 'PRIN1)
 A B C NIL
 ---------- ---------- ---------- ---------- ----------
(MAP '(A B C) 'PRIN1)
 (A B C) (B C) (C) NIL
 ---------- ---------- ---------- ---------- ----------
(AND (PRIN1 1) (PRIN1 2) (PRIN1 3))
 1 2 3 3
 ---------- ---------- ---------- ---------- ----------
(AND (PRIN1 1) NIL (PRIN1 2))
 1 NIL
 ---------- ---------- ---------- ---------- ----------
(OR (PRIN1 1) (PRIN1 2) (PRIN1 3))
 1 1
 ---------- ---------- ---------- ---------- ----------
(OR NIL NIL (PRIN1 1))
 1 1
 ---------- ---------- ---------- ---------- ----------
(SETQ L '(()))
 (NIL)
 ---------- ---------- ---------- ---------- ----------
(SETQ X 5)
 5
 ---------- ---------- ---------- ---------- ----------
(WHILE (GTZ X)
       (NCONC1 L (READCH))
       (SETQ X (SUB1 X)))BRAVO
 NIL
```

```
---------- ---------- ---------- ---------- ----------
(CDR L)
 (B R A V O)
---------- ---------- ---------- ---------- ----------
(GENSYM)
 G0001
---------- ---------- ---------- ---------- ----------
(GENSYM)
 G0002
---------- ---------- ---------- ---------- ----------
(GENSYM)
 G0003
---------- ---------- ---------- ---------- ----------
(OBLIST)
 (G0003 G0002 G0001 V R ONE DEUX UN HOP L D C A N O B ).(/ X ;;;; SAVE
---------- T1600 LISP TEST)
---------- ---------- ---------- ---------- ----------
(LENGTH (OBLIST))
 24
---------- ---------- ---------- ---------- ----------
(REVERSE '(1 2 3))
 (3 2 1)
---------- ---------- ---------- ---------- ----------
(SUBST 'A 1 '(1 (1 2 1) 1 .1))
 (A (A 2 A) A . A)
---------- ---------- ---------- ---------- ----------
(SUBST '(A A) 1 '(1 (1 2 1) 1 .1))
 ((A A) ((A A) 2 (A A)) (A A) A A)
---------- ---------- ---------- ---------- ----------
(MEMQ 'B '(A B C D))
 (B C D)
---------- ---------- ---------- ---------- ----------
(MEMQ 'E '(A B C D))
 NIL
---------- ---------- ---------- ---------- ----------
(MEMQ 3 '(1 2 3 E 4 5))
 (3 E 4 5)
---------- ---------- ---------- ---------- ----------
(SETBIT 12)
 000CHH
---------- ---------- ---------- ---------- ----------
(LOGAND 0F00HH 5555HH)
 0500HH
---------- ---------- ---------- ---------- ----------
(LOGOR FF00HH 00FFHH)
 FFFFHH
---------- ---------- ---------- ---------- ----------
(LOGSHIFT 0FF0HH 4)
 FF00HH
---------- ---------- ---------- ---------- ----------
(LOGSHIFT 0FF0HH -4)
 00FFHH
---------- ---------- ---------- ---------- ----------
(CLRBIT 12)
 12
---------- ---------- ---------- ---------- ----------
(SETQ L '(A B))
 (A B)
---------- ---------- ---------- ---------- ----------
(APPEND L '(C D))
 (A B C D)
---------- ---------- ---------- ---------- ----------
```

```
(APPEND L 'C)
 (A B . C)
---------- ---------- ---------- ---------- ----------
(NCONC L '(C D))
 (A B C D)
---------- ---------- ---------- ---------- ----------
L
 (A B C D)
---------- ---------- ---------- ---------- ----------
(NCONC1 L 'E)
 (A B C D E)
---------- ---------- ---------- ---------- ----------
L
 (A B C D E)
---------- ---------- ---------- ---------- ----------
(NCONC1 L '(F G))
 (A B C D E (F G))
---------- ---------- ---------- ---------- ----------
L
 (A B C D E (F G))
---------- ---------- ---------- ---------- ----------
(EQUAL 1 1)
 T
---------- ---------- ---------- ---------- ----------
(EQUAL 'A 'A)
 T
---------- ---------- ---------- ---------- ----------
(EQUAL 1 5)
 NIL
---------- ---------- ---------- ---------- ----------
(EQUAL 'A 'B)
 NIL
---------- ---------- ---------- ---------- ----------
(EQUAL '(A . B) '( A . B))
 T
---------- ---------- ---------- ---------- ----------
(EQUAL (OBLIST) (OBLIST))
 T
---------- ---------- ---------- ---------- ----------
(EQUAL '(A (B . C)) '(A (B . D)))
 NIL
---------- ---------- ---------- ---------- ----------
(SETQ L '((A B C) (D E F) (G H I) J))
 ((A B C) (D E F) (G H I) J)
---------- ---------- ---------- ---------- ----------
(CAR L)
 (A B C)
---------- ---------- ---------- ---------- ----------
(CDR L)
 ((D E F) (G H I) J)
---------- ---------- ---------- ---------- ----------
(CAAR L)
 A
---------- ---------- ---------- ---------- ----------
(CDAR L)
 (B C)
---------- ---------- ---------- ---------- ----------
(CADR L)
 (D E F)
---------- ---------- ---------- ---------- ----------
(CDDR L)
 ((G H I) J)
```

```
------------ ------------ ------------ ------------ ------------
(CADDR L)
 (G H I)
------------ ------------ ------------ ------------ ------------
(CADAR L)
 B
------------ ------------ ------------ ------------ ------------
(CAADR L)
 D
------------ ------------ ------------ ------------ ------------
(CADADR L)
 E
------------ ------------ ------------ ------------ ------------
(CAADDR L)
 G
------------ ------------ ------------ ------------ ------------
(CADADDR L)
 H
------------ ------------ ------------ ------------ ------------
(CADDDR L)
 J
------------ ------------ ------------ ------------ ------------
(SET 'A 'B)
 B
------------ ------------ ------------ ------------ ------------
A
 B
------------ ------------ ------------ ------------ ------------
(LISTP 1)
 NIL
------------ ------------ ------------ ------------ ------------
(LISTP 'A)
 NIL
------------ ------------ ------------ ------------ ------------
(LISTP '(A B))
 T
------------ ------------ ------------ ------------ ------------
(SETQ L '((A . 1) (B . 2) (C . 3)))
 ((A . 1) (B . 2) (C . 3))
------------ ------------ ------------ ------------ ------------
(ASSQ 'A L)
 (A . 1)
------------ ------------ ------------ ------------ ------------
(CASSQ 'A L)
 1
------------ ------------ ------------ ------------ ------------
(ASSQ 'D L)
 NIL
------------ ------------ ------------ ------------ ------------
(CASSQ 'D L)
 NIL
------------ ------------ ------------ ------------ ------------
(SET (ASSQ 'C L) 'D)
 D
------------ ------------ ------------ ------------ ------------
L
 ((A . 1) (B . 2) (D . 3))
------------ ------------ ------------ ------------ ------------
(RPLACA 'CAR CDR (SETQ X CAR))
 CAR
------------ ------------ ------------ ------------ ------------
(CAR '(A B C))
```

```
 (B C)
 ________ ________ ________ ________ ________
(RPLACA 'CAR X)
 CAR
 ________ ________ ________ ________ ________
(CAR ' (A B C))
 A
 ________ ________ ________ ________ ________
(DE SORT L
       (MAP L '(LAMBDA (X) (AND (CDR X)
                               (MAP (CDR X) '(LAMBDA (Y ;; AUX)
                                    (AND (GT (SETQ AUX (CAR X)) (CAR Y))
                                         (SET X (CAR Y))
                                         (SET Y AUX)))))))
       L)
 SORT
 ________ ________ ________ ________ ________
(SORT 10 9 7 8 6 5 3 4 1 2)
 (1 2 3 4 5 6 7 8 9 10)
 ________ ________ ________ ________ ________
(PROG (X Y)
        (SETQ X 3)
       (WHILE  (GTZ X)
               (SETQ Y 2)
               (WHILE (GTZ Y)
                      (PRIN1 X) (PRINT Y)
                      (SETQ Y (SUB1 Y)))
               (SETQ X (SUB1 X))))
 3 2
 3 1
 2 2
 2 1
 1 2
 1 1
 NIL
 ________ ________ ________ ________ ________
(DE HANOI (N DEPART ARRIVEE INTERMEDIAIRE)
       (COND
               ((GTZ N)
                (HANOI (SUB1 N) DEPART INTERMEDIAIRE ARRIVEE)
                (MAPC (LIST 'DISQUE N 'DE DEPART 'VERS ARRIVEE) 'PRIN1)
                (TERPRI)
                (HANOI (SUB1 N) INTERMEDIAIRE ARRIVEE DEPART))))
 HANOI
 ________ ________ ________ ________ ________
(HANOI 1 'A 'C 'B)
 DISQUE 1 DE A VERS C
 NIL
 ________ ________ ________ ________ ________
(HANOI 2 'A 'C 'B)
 DISQUE 1 DE A VERS B
 DISQUE 2 DE A VERS C
 DISQUE 1 DE B VERS C
 NIL
 ________ ________ ________ ________ ________
(HANOI 3 'A 'C 'B)
 DISQUE 1 DE A VERS C
 DISQUE 2 DE A VERS B
 DISQUE 1 DE C VERS B
 DISQUE 3 DE A VERS C
 DISQUE 1 DE B VERS A
 DISQUE 2 DE B VERS C
```

```
 DISQUE 1 DE A VERS C
 NIL
 ----------------------------------------------------------
(HANOI 4 'A 'C 'B)
 DISQUE 1 DE A VERS B
 DISQUE 2 DE A VERS C
 DISQUE 1 DE B VERS C
 DISQUE 3 DE A VERS B
 DISQUE 1 DE C VERS A
 DISQUE 2 DE C VERS B
 DISQUE 1 DE A VERS B
 DISQUE 4 DE A VERS C
 DISQUE 1 DE B VERS C
 DISQUE 2 DE B VERS A
 DISQUE 1 DE C VERS A
 DISQUE 3 DE B VERS C
 DISQUE 1 DE A VERS B
 DISQUE 2 DE A VERS C
 DISQUE 1 DE B VERS C
 NIL
 ----------------------------------------------------------
(PROG ()
       (ESCAPE H1 (PRINT 1))
       (PRINT 2))
 1
 2
 2
 ----------------------------------------------------------
(PROG ()
       (ESCAPE H1 (PRINT 1) (H1 (PRINT 2))))
 1
 2
 2
 ----------------------------------------------------------
(SETQ X 3)
 3
 ----------------------------------------------------------
(ESCAPE H1
       (PROG ()
         A     (ESCAPE H2 (OR (GTZ X) (H1))
                          (PRINT 0) (H2))
               (PRINT X)
               (SETQ X (SUB1 X))
               (GO A)))
 0
 3
 0
 2
 0
 1
 NIL
 ----------------------------------------------------------
(ESCAPE H1
       (ESCAPE H2
               (ESCAPE H3
                       (ESCAPE H4 (H3)))
               (PRINT 1) (H1 2))
       (PRINT 3))
 1
 2
 ----------------------------------------------------------
(SETQ X 3)
```

```
3
------------ ------------ ------------ ------------ ------------
(ESCAPE H1
        (PROG ()
          A      (ESCAPE H2
                          (PROG ()
                               (ESCAPE H3
                                         (PROG ()
                                                (ESCAPE H4
                                                        (PROG () (H3))))))
                               (PRINT 1))
                          (PRINT 2) (H2))
                 (AND (GTZ X) (SETQ X (SUB1 X)) (GO A)))
        (PRINT 3))
 1
 2
 1
 2
 1
 2
 1
 2
 3
 3
------------ ------------ ------------ ------------ ------------
(DE FORMAT (L NC NAL ;; X N)
        (SETQ X 1) (SETQ NN NAL)
        (WHILE L
                (TTAB X)
                (PRIN1 (NEXTL L)) (SETQ X (PLUS X NC))
                (SETQ NN (SUB1 NN))
                (COND ((ZEROP NN) (TERPRI) (SETQ NN NAL) (SETQ X 1))))
        (OR (EQ NN NAL) (TERPRI))
        T)
 FORMAT
------------ ------------ ------------ ------------ ------------
(PROGN (SETQ RES '(NIL)) (SETQ X 6)
        (WHILE (LT (SETQ Y (PLUS (LOC NIL) X)) (ADD1 (LOC 'SYS3)))
               (NCONC1 RES (VAG Y)) (SETQ X (PLUS X 6)))
        (FORMAT RES 10 7))
 NIL       QUOTE     '         EVAL      APPLY     EVLIS     GOTO
 RETURN    READ      ATOM      EPROGN    RPLACA    RPLACD    CONS
 PRINT     PRIN1     TERPRI    OBLIST    NOT       NULL      EQ
 NEQ       GTZ       CAR       CDR       ADD1      SUB1      SWITCH
 RESET     GET       PUT       SETBIT    CLRBIT    TIMES     PLUS
 ZEROP     DIFFER    QUO       LOC       REM       NUMBP     GT
 GE        LT        LE        LOGAND    LOGOR     LOGSHIFT  LENGTH
 NTH       VAG       APPEND    EQUAL     CAAR      CADR      CDAR
 CDDR      CADDR     CADDDR    CAADR     CADADR    CAADDR    CADADDR
 SET       LISTP     NCONC1    NCONC     MEMQ      MAP       MAPC
 READCH    SPACES    TTAB      GENSYM    PAGE      CASSQ     ASSQ
 REVERSE   SUBST     CADAR     LIST      DE        DF        PROGN
 NEXTL     SETQ      WHILE     PROG      GO        ESCAPE    ESCZZZZZ
 MCHAR     OR        AND       OPENOLD   COND      T         EXPR
 FEXPR     LAMBDA    SYS1      SYS2      SYS3
 T
------------ ------------ ------------ ------------ ------------
(MCHAR $ (LAMBDA (;; RES X)
                (SETQ RES '(()))
                (WHILE (NEQ (SETQ X (READCH)) '$)
                       (NCONC1 RES X))
                (CDR RES)))
```

```
$
-------- -------- -------- -------- --------
'$JE SUIS TRISTE$
 (J E   S U I S   T R I S T E)
-------- -------- -------- -------- --------
(OBLIST)
 (U S   $ RES NN NAL NC FORMAT H4 H3 H2 H1 VERS DISQUE INTERMED
ARRIVEE DEPART HANOI AUX Y SORT J I H G F E G0003 G0002 G0001 V R ONE
DEUX UN HOP L D C A N O B ).(/ X ;;;; SAVE ________ T1600 LISP TEST)
-------- -------- -------- -------- --------
(NULL (RPLACA (CDR 'ATOM) SAVE))
 NIL
-------- -------- -------- -------- --------
(OBLIST)
 (SAVE ________ T1600 LISP TEST)
-------- -------- -------- -------- --------
(RESET '(FIN DU TEST LISP T1600))
```



UTILITAIRES

PRETTY-PRINT : édite des fonctions LISP sous forme lisible et justifiée.

<u>Utilisation</u> :

(PRETTY nom_1 nom_2 ..nom_n)

les $noms_i$ étant les noms de fonctions à éditer.

TRACE : permet de tracer les appels de fonctions de diverses façons.

<u>Utilisation</u> :

(TRACE nom_1 nom_2 ... nom_n)

Tous les appels des fonctions nommées seront imprimés ainsi que la valeur de leurs arguments. Au retour seront également imprimés les noms des fonctions ainsi que leur valeur.

Pour les "détracer" faire
(UNTRAC nom_1 nom_2 ... nom_n)

Pour tracer, dans des fonctions toutes les affectations par SETQ sous forme :
variable = valeur affectée
faire (TRACEQ nom_1 ... nom_n)
et (UNTRACQ nom_1 ... nom_n) pour les détracer.

Pour tracer, dans des PRØGs tous les branchements par GØ sous forme :
(ETIQ : étiquette de branchement)
faire (TRACEGØ nom_1 ... nom_n) et (UNTRACG ...)
pour les détracer.

LAP : assemble et charge des fonctions LISP rédigées en LAP (PROGRAMME ASSEMBLEUR LISP) pour T1600.

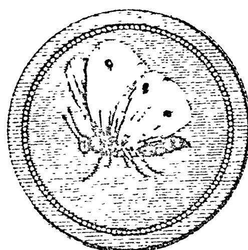
<u>Utilisation</u> : pour les SUBR faire (LAP nom DE)
pour les FSUBR faire (LAP nom DF)

Un exemple est donné : l'assemblage de la fonction PREPARER, de l'interprète CØNNIVER-T1600.

LISTING DE TRACE

```
(DF TRACE (%L)
      (MAPC %L '(LAMBDA (%X1)
                 (SET '%X2 (OR (GET %X1 EXPR) (GET %X1 FEXPR)))
                 (OR (GET %X1 'PRINTL) (PUT %X1 %X2 'PRINTL))
                 ((LAMBDA (%X1 %X2)
                     (COND4
                        ((GET %X1 EXPR) (PUT %X1 %X2 EXPR))
                        ((GET %X1 FEXPR) (PUT %X1 %X2 FEXPR))))
                  %X1 (LIST (CAR %X2) (CADR %X2)
                        (LIST 'PRTRAC (CADR %X2) %X1))))))
(DF PRTRAC (%L %X1 %X2)
         (SET '%X1 (CAR %L))
         (SET '%X2 (CADR %L))
      (PRINTL '-----) (COND ((ATOM %X1) (EVAL %X1))
                              ((EVLIS %X1))))
      (SET '%X1 (EVAL  (CADDR (GET %X2 'PRINTL))))
      (PRINTL '(----- %X1)
      %X1)
(DE PRINTL (%X1 %X3)
      (PRIN1 %X1) (PRIN1 %X2) (PRIN1 '/ / ) (PRINT %X3))
(DF UNTRAC (%L)
      (MAPC %L
                 '(LAMBDA (%L)
                 (RPLACA (CDDR %L) (GET %L 'PRINTL))
                 (RPLACD (CDDR %L)))))
(DE %TRTF (%X1 %X2)
         (MAPC %L '(LAMBDA  (%L)
                    (RPLACD %L (SUBST %X1 %X2 (CDR %L))))))

(DF TRACEQ (%L) (%TRTF '%TSETQ 'SETQ))
(DF UNTRACQ (%L) (%TRTF 'SETQ '%TSETQ))
(DF TRACEGO (%L) (%TRTF '%TGO 'GO))
(DF UNTRACG (%L) (%TRTF 'GO '%TGO))
(DF %TSETQ (%L)
         (SET (PRIN1 (CAR %L)) (PROGN
                (PRIN1 '=) (PRINT (EVAL (CADR %L))))))
(DF %TGO (%L)
         (PRINT (CONS 'ETIQ: %L))
         (GOTO (CAR %L)))
```



LISTING DU PRETTY-PRINT

```
(DF PRETTY (L I LESCAPE XPRTTY)
   (SETBIT 15)
   (SETQ XPRTTY '(LAMBDA ()  (SETQ I (ADD1 I))
                             (MAPC E '(LAMBDA (E)
                               (ENDLINE) (SUPERPRINT E) ))
                             (SETQ I (SUB1 I))
                             (PRIN1 '/))))
   (MAPC L
      '(LAMBDA (L) (TERPRI) (PRIN1 '/() (PRINT L)
                   (TTAB (SETQ I 3))
                   (SUPERPRINT (CADDR L)) (PRIN1 '/))))
       (CLRBIT 15) (TERPRI))

(DE SUPERPRINT (E L)
   (ESCAPE EXIT
    (COND
     ((ATOM E) (PRIN1 E))
     ((EQ (CAR E) QUOTE) (PRIN1 '/') (SUPERPRINT (CADR E)))
     ((PRIN1 '/()
      (WHILE E (SETQ L (NEXTL E))
         (COND
          ((NULL E))
          ((OR (MEMQ L LESCAPE)
               (MEMQ L '(OR AND LIST PLUS TIMES WHILE PROGN MAP MAPC)))
                  (PRIN1 L)
           (EXIT (XPRTTY)))
          ((EQ L 'ESCAPE)      (PRIN1 L)
           (SETQ LESCAPE (CONS (CAR E) LESCAPE))
           (EXIT (XPRTTY) (NEXTL LESCAPE)))
          ((EQ L LAMBDA)
           (PRIN1 LAMBDA) (SPACES 1)
           (SUPERPRINT (NEXTL E)) (EXIT (XPRTTY)) )
          ((EQ L 'COND)
           (PRIN1 L) (SETQ I (ADD1 I))
           (MAPC E
            '(LAMBDA (E) (ENDLINE) (PRIN1 '/()
                            (SUPERPRINT (NEXTL E))
             (XPRTTY) ))
           (SETQ I (SUB1 I))
           (EXIT (PRIN1 '/))))
          ((EQ L 'PROG)
           (PRIN1 L) (SPACES 1) (SUPERPRINT (NEXTL E))
           (WHILE E
            (ENDLINE)
            (COND
             ((ATOM (CAR E))
              (PRIN1 (NEXTL E)) (TTAB (TIMES 3 (PLUS I 2))) )
             ((SPACES 6)))
            (SETQ I (PLUS I 2)) (SUPERPRINT (NEXTL E))
            (SETQ I (DIFFER I 2)))
           (SETQ I (SUB1 I))
           (EXIT (PRIN1 '/)))))
        (SUPERPRINT L) (AND E (SPACES 1)))
       (PRIN1 '/))))))

(DE ENDLINE () (TERPRI) (TTAB (TIMES I 3)))
```

EXEMPLE DE LAP

```
(LAP PREPARER DE)
          ($ PREPARER)                        (CDR PREPARER)
          (LA A1)                             50FAHH
          (LR A L)                            2BC5HH
          (CP NATOM)                          5503HH
          (JLE FINPREP)                       0700HH
          (LA CAR L)                          9000HH
          (CP ADNUMB)                         5541HH
          (JE FINPREP)                        0600HH
          (CP (QUOTE ÷))                      (CAR 5500HH ÷)
          (JNE PREP2)                         0200HH
          (LA CDR L)                          9001HH
PREP0     (STA A1)                            4DFAHH
          (RSR)                               1E02HH
PREP2     (CP (QUOTE ,))                      (CAR 5500HH ,)
          (JNE PREP4)                         0200HH
PREP3     (LA & CDR L)                        B001HH
          (STA A1)                            4DFAHH
          (RSR)                               1E02HH
PREP4     (CP (QUOTE @))                      (CAR 5500HH @)
          (JNE PREP6)                         0200HH
          (LA CDR L)                          9001HH
          (STA A1)                            4DFAHH
          (BR EVALP)                          45E7HH
PREP6     (CP (QUOTE $))                      (CAR 5500HH $)
          (JNE PREP5)                         0200HH
          (LA & CDR L)                        B001HH
          (CP (QUOTE UNASSIGN))               (CAR 5500HH UNASSIGN)
          (JNE PREP0)                         0200HH
          (RSR)                               1E02HH
PREP5     (STA A1)                            4DFAHH
          (LR K A)                            2BF8HH
          (ADRI 0002HH A)                     0802HH
          (CP TSTACK)                         5508HH
          (JL $ 0002HH)                       0502HH
          (BR ERRFS)                          451BHH
          (LB CDR L)                          9701HH
          (PSR B)                             1A40HH
          (BSR (PREPARER))                    (CAAR 4600HH PREPARER)
          (LB A1)                             57FAHH
          (PLR A)                             1B01HH
          (PSR B)                             1A40HH
          (STA A1)                            4DFAHH
          (BSR (PREPARER))                    (CAAR 4600HH PREPARER)
          (PLR A)                             1B01HH
          (LB A1)                             57FAHH
          (BSR CONSP)                         46DDHH
          (STA A1)                            4DFAHH
FINPREP   (RSR)                               1E02HH
```

EXEMPLE DE GENERATION DE PLAN

EN CONNIVER T1600.

```
(SETQ CØNTEXT NIL)

(MAPC '((ØN 1 TABLE) (ØN 2 1) (ØN 3 2)
        (ØN 4 5) (ØN 5 6) (ØN 6 TABLE)) 'ADD)

(DE FREE (X ;; Z)
      (AND (PRESENT '(ØN !Z ,X))
           (FREE1 Z)))

(DE FREE1 (X)
      (FREE X)
      (ADD '(ØN ,X TABLE)))

(DE SCENE (;; X Y)
      (FØR-EACH (ØN !X !Y) (PRINT CURRENT)))

(ADD (IF-ADDED IF-ØN (ØN !X !Y)
      %AUX% (X Y Z)
      (REMØVE (PRESENT '(ØN ,X !Z)))))

(ADD (IF-ADDED IF-P-ØN (IF-PUT-ØN !X !Y)
      %AUX% (X Y)
      (FREE X)
      (FREE Y)
      (ADD '(ØN ,X ,Y)))))

(ADD (IF-REMØVED RMVD(ØN !X !Z)
      %AUX% (X Z)
      (PUSH-PLAN 'REMØVE =(ØN ,X ,Z))
      (PUSH-PLAN 'ADD =(ØN ,X ,Y)))))

(DE PUSH-PLAN (INDIC WHAT)
      (SETQ PLAN (NCØNC1 PLAN
                        (LIST INDIC (LIST 'QUØTE WHAT)))))
```

```
;       ****** ****** ****** A C T I Ø N ****** ****** ******;
(SETQ PLAN NIL)
 NIL
(SCENE)
 (ØN 6 TABLE)
 (ØN 5 6)
 (ØN 4 5)
 (ØN 3 2)
 (ØN 2 1)
 (ØN 1 TABLE)
 NIL

(ASSUMING (IF-PUT-ØN 5 2) (SCENE))
 (ØN 5 2)
 (ØN 3 TABLE)
 (ØN 4 TABLE)
 (ØN 6 TABLE)
 (ØN 2 1)
 (ØN 1 TABLE)
 T

(SCENE)
 (ØN 6 TABLE)
 (ØN 5 6)
 (ØN 4 5)
 (ØN 3 2)
 (ØN 2 1)
 (ØN 1 TABLE)
 NIL

(MAPC PLAN 'PRINT)
 (REMØVE (QUØTE (ØN 4 5)))
 (ADD (QUØTE (ØN 4 TABLE)))
 (REMØVE (QUØTE (ØN 3 2)))
 (ADD (QUØTE (ØN 3 TABLE)))
 (REMØVE (QUØTE (ØN 5 6)))
 (ADD (QUØTE (ØN 5 2)))
 NIL
```

